Data Mining Application in a Software Project Management Process

Richi Nayak[1], and Tian Qiu[2]

[1]School of Information Systems, QUT, Brisbane, QLD 4001, Australia
r.nayak@qut.edu.au
[2]EDS Credit Services, Adelaide, Australia, tian.qiu@eds.com

Abstract*: During the life cycle of a software project development, many problems are found and raised. Resolutions to these problems are very time consuming and costly. How to use data mining techniques to analyse these problems, and find valuable knowledge to reduce the effort of fixing these problems are discussed in this paper.*

1 Introduction

A project leader manages a project with several issues involved such as project planning and scheduling, code implementation, testing and release. It is difficult for him to precisely estimate the project duration in advance. However, he can utilise the Data Mining (DM) methods and make the accurate estimation by learning from the information gained by previous projects. He can also eliminate several potential problems when a pattern appears in his current project similar to the one that have caused problems in previous projects.

Data mining techniques have been successfully applied to various areas such as marketing, medical, and financial [4, 5, 6, 7]. However, few of them can be currently seen in software engineering domain. There exist several difficulties, such as hard to find a data model to put through mining process, no suitable mining tools, poor data quality and acquisition etc. This paper explores the software engineering domain by applying DM techniques to a real world data set. This paper first introduces the undertaken software engineering problem and data mining. After a data model is established for the underlying problem, various data mining techniques are experimented. Interesting findings are discussed together with issues appearing during the mining process.

2 Data Acquisition

Every year, more than 50 software projects are carried out in MASC[*], more than ten thousands lines of code are created, and thousands pages of documents are released to various customers. In order to control the problems appearing during a project life

[*] MASC is a division of a global telecommunication company. A detail of the information source is intentionally removed.

cycle and to improve the working efficiency, MASC has set up a series of processes such as Software Configuration Management, Software Risk Management, Software Project Metric Report and Software Problem Report Management. Data is collected during all these procedures. The Software Problem Report (PR) management data is chosen as the focus of this research. A software bug-tracking system, GNATS (**A** **T**racking **S**ystem by **GN**U), is set up on MASC Intranet to collect and maintain all PRs raised from every department and individual within MASC. Currently the GNATS system stores more than 40,000 of problem reports.

Each PR page starts with a number together with a series of information about when, who and which project or department raised this PR. After this, there are several fields that give details of the PR such as *Synopsis, Severity, Priority, Responsible, State, Class, Arrival-Date, Closed Date, Description, etc*. *Description* details about the bug and its affect the whole system. *Synopsis* is a summary of *Description* field. The *Severity* field shows the criticality of the problem as - *serious*, *critical* or *non-critical*. The *Priority - high, medium or low* – field shows how soon the problem should be resolved. The *State* attribute tells the current stage in the progress of the PR. User can only choose from *open, active, analysed, suspended, feedback, resolved and closed* as the input value of this field. Another important variable is *Class* to state what type the bug is - *sw-bug, doc-bug, change-request, duplicate, mistaken,* or *support.*

3 Data Pre-processing

A data mining task includes the preprocessing of data before valid, novel, potentially useful, and ultimately understandable patterns are identified in data.

3.1 Defining Goals

MASC engineers make estimations on many aspects of a project such as the number of lines of code to be developed, the kinds of document to be delivered to customer, the time required to accomplish each software engineering stage in the project. There are several tools exist to help a programmer to do the implementation jobs in the software design stage. However, there is few or even no tool exists that can be used for both estimation and project problem reasoning stage. Project team members can only give estimations based on their own experience from previous projects. If the current project is not within their familiar topics, the accuracy of the estimation becomes worse. A PR (problem report) fixing work also becomes more tedious when the responsible person can not estimate the time to fix the problem. Finding a precise estimation figures on bug fixing or estimation work at the early stage of a project will bring great cost savings and accurate progress control to the development team and to the organization.

Currently the GNATS system has no actual database management system implemented. If a PR is closed, it is just statically stored in GNATS and no further analysis is performed. This limits the potential benefits to software engineers who can obtain valuable information if the existing PR data is being analysed. This is useful especially when a programmer is struggling with a bug while a resolution may already hide behind the knowledge that can be derived from the previous similar problems.

These problems can be relieved by utilising data mining techniques. Mining results of the PR data will bring benefits such as accurate project estimation and planing, improved control over the PR fixing and deduction of the project cycle time.

3.2 Field Selection

There are several fields such as *Confidential, Submitter-ID, Environment, Fix, Release Note, Audit Trail, the associated project name and PR number* that are ignored during mining. These fields provide identification information about a PR containing no mining value. Instead, some of these values are used as support roles during pre-processing and post-processing stages to assist in the selection of data and a better understanding of the rules being found.

The aim of this mining exercise is to find useful knowledge from existing projects, all the existing projects should already be finished and all the corresponding PRs should also be closed. Otherwise, a PR can still be changed and is not stable for mining. Hence all PRs with a ***closed*** value in their *State* field are chosen.

Whenever a PR is raised, a project leader will have to find answers for the following questions before taking any action: How long it will take to fix? How many people were involved? How severe the problem is (customer impact)? What is the impact of the problem on project schedule (Cost & Team priority)? and What type of the problem it is (a Software bug or a design flaw)? Accordingly, attributes such as *Severity, Priority, Class, Arrival-Date, Closed-Date, Responsible,* and *Synopsis* are considered for mining. The first three attributes describe how a PR is handled within a project, the next three attributes indicate how long a PR is fixed and who were responsible, and the last one lists the content in a PR. The attribute '*class*' is chosen as the target attribute in order to find out any valuable knowledge among the type of a problem and the rest of the PR attributes. Knowing the relationship between the fix effort (in time) and the PR class, a project leader can analyse the fix effort versus the human resources available, and put it in the schedule and resource plan.

The first five fields have fixed input values. *Responsible* attribute is used to calculate how many people were involved to fix the problem. Association or characteristics rules can be found by applying mining techniques on these fields. Whereas *Synopsis* field has no fixed input values. Instead, there is a lot of pure text information stored in this field briefly describing what the problem is in the associated project. It may contain what type of a project document (a piece of code or a support document) that the PR is concerned with. It can be used as a text index. Due to the nature of the values inside this field, a different mining approach, Text Mining is considered to deal with such kind of text values.

3.3 Data Cleaning

Data is further investigated to identify problems, such as missing values, inconsistent values, and mistaken values using graphical tools such as histogram for frequency distribution of the values, calculating maxima, minima and mean values. Histogram plots the contribution made by each value for the (categorical) attribute, and therefore helps to identify distribution skews and invalid values. The occurrence of these problems comes with several factors such as human mistakes and evolutions of the

GNATS system. An example is the use of different terminologies over the time such as *SW-bug* or *sw-bug* as an input value for *Class* field (Example a, d in Figure 1). A *Time-Zone* field and other new input values have been added later in the system on management request based on feedback of users after several years of system running.

For the PRs in which an error can be recovered manually or automatically by software, the modified PRs are included in the mining process. For example, *SW-bug* in *Class* field is replaced by *sw-bug* throughout the data. Another example is filling the data in wrong fields such as mistakenly input the closing date in the obsolete *Completed-Date* field instead of the *Closed-Date* field.

The PRs, in which an error cannot be recovered precisely, are either discarded or replaced by a '?' if a software can handle the missing values. For example, the pattern a in Figure 1 has its closed time earlier than the time being raised. Some PRs do not have all the values stored, such as Example **c** in Figure 1 has no closed date. An example of inconsistent values is shown in Figure 1 - there is no input for the *Time-Zone* field in a PR recorded before 1998, as the *Time-Zone* field is added in 1998.

PR_ID|*Category*|*Severity*|*Priority*|*Class*|*Arrival-Date*|*Close-Date*|*Synopsis*

a. 17358|bambam|serious|high|*sw-bug*|*20:50 May 25 CST 1999*|*11:35 Mar 24 CST 1999*| STI STR register not being reset at POR
b. 17436|bambam|serious|high|support|18:10 Mar 30 CST 1999|12:00 May 24 CST 1999| sequence_reg varable in the RDR_CHL task is not defined
c. 580 |bingarra|serious|low|doc-bug|10:10 May 31 May 1996| | In URDRT2 of design doc, the word 'last' should be 'first'
d. 6205 |gali|serious|medium|*SW-bug*|14:30 Nov 5 1997|13:14 Dec 1 1997| grouping of options in dialog box

Figure 1: Data examples from the original PR data set

Severity |*Priority*| *Time-to-fix*| *Class* |*Synopsis*

a. serious, high, 61, sw-bug, STI STR register not being reset at POR
b. serious, high, 56, support, sequence_reg varable in the RDR_CHL task is not defined
c. serious, low, ?, doc-bug, In URDRT2 of design doc, the word 'last' should be 'first'
d. serious, medium, 24, sw-bug, grouping of options in dialog box

Figure 2: Data examples ready for mining

3.4 Data Transformation

Data transformation is considered, in the way of converting attributes *Arrival-Date* and *Closed-Date* to a time-period - identifying the time spent to fix a PR - taking account the additional information *Time-Zone* and *Responsible*. This transformation resulted in the *Time-to-fix* attribute with continuous values (figure 2). The *Responsible* attribute

has the information about personnel engaged in rectifying the problem. We assume that the derived attribute *Time-to-fix* is total time spent to fix a problem if there is only one person involved. The calculated time period from *Arrival-Date* and *Closed-Date* is than multiplied by the number of people yield from the *Responsible* attribute. In order to improve the mining quality, we have discretized this attribute with cutting points be one day (1), half week (3 days), one week (7), two weeks (14), one month (30) and one quarter (90 days), half year (180 days) and more than one year (360 days). So that the mining results are not very highly depended on the exact human resource involved but gives an approximate estimate, allowing a minor change in human resource.

4 Data Modelling and Mining

Out of total 40000 PRs initially selected as the data set, we are left with 11,000 PRs after pre-processing (figure 2). These 11,000 PRs (depends on different mining tasks, the numbers varies a little) cover more than 120 projects within MASC from 1996 to 2000. For example 11364 PR records have been applied with text-mining tools on the valid values in their Synopsis fields, as 364 records have no time values so could not be used for classification task.

The GNATS system provides simple methods to retrieve basic information from the PR data set, such as the PR numbers related to a particular person, etc. Besides this, general database query languages (such as SQL) can also give useful information, i.e., the average time spent for fixing a PR in a project. However these methods cannot perform if user likes to (1) pose queries on a large number of records with high dimensional structures, (2) summarise a large data set to facilitate decision-making, (3) make predictions on new data based on the existing rules, and (4) visualise simplified extracted local structures. On the contrary, data mining techniques perform well on these cases, and are able to reveal the deeper characters of the data, such as:

- If a PR is raised, how long should it take to fix the problem?
- What type of project documents needs a significant effort to fix an associated bug?

The selection of data mining operations and techniques is one of the most important things that directly affect the progress and the accomplishment of any DM applications. We have chosen:

- Prediction modelling on the time consuming patterns of the PR data to make estimation.
- Link analysis to discover association among the contents/values of the variables being selected.
- Text Mining to *analyse Synopsis* field.

The predictive modelling or classification task builds a model on existing dataset by recognising distinct characteristics of the data set. The built model predicts future events based on previous data, specifying a class (or label) to each record in the dataset. A supervised machine learning algorithm, that learns a model on previous or existing data, can be used for predictive modelling task. These models are developed over training and testing phases. The model is given some already known facts with correct answers during the training phase, from which the model learns to make accurate predictions. During the testing phase, the model is exposed to new data set to check the predictive capability. Various classification methods are neural induction,

tree induction and bayesian classifiers, K-nearest neighbour classifiers, case based reasoning, genetic algorithms, rough set and fuzzy set approaches [4, 5, 6, 7].

Tree induction or decision tree is used for this task. Decision tree has been quite popular in data mining due to their simplicity, efficiency and capability of dealing with a large number of training examples. The decision tree learning algorithms start by constructing a decision tree from top to bottom. Attributes are evaluated at each step to form descendant nodes. The attribute selection is based on a `statistical test' to determine how well it classifies the training examples. An internal node represents a test on an attribute and a leaf node represents a class or class distribution. Classification of unknown samples is made by tracing a path through the decision tree until a leaf node having the class prediction is reached. The maximum height of the decision tree depends on the number of attributes used to define the rules. Because the number of attributes in our problem is small, the resulting decision tree is relatively simple and thus its structure is understood easily by a human analyst.

The link analysis operation exposes samples and trends by predicting correlation of variables in a given data set. Association discovery builds a model to find variables implying the presence of other variables (with a certain degree of confidence and support) in the given data set. This process reveals hidden affinity among the variables i.e. which variables cause one another if a PR report is being raised. The technique is based on counting occurrences of all possible combination of variables. Apriori and its variation algorithms [6] are most widely used.

In general, Data mining is knowledge discovery from structured databases. Text mining techniques, on the other hand, discover the knowledge from an unstructured textual data. In order to discover and use the implicit structure of the texts, text mining techniques integrate some specific natural language processing to pre-process the textual data. A suitable data-mining tool should satisfy the ease of use, low cost, ease of preparation and appropriate for the data model [5]. The C5 [2] is used for classification, CBA [1] is used for both classification and association rules mining, and TextAnalyst [3] is used for text mining.

5 Assimilation and Analysis of Outputs

5.1 Classification and Association Rule Mining

In order to get better rules and to decrease the error rate as much as possible, several approaches are used. One approach is to stratify the data on the target using the choice-based sampling rather than using random samples. Equal number of samples representing each possible value of the target attribute (*Class*) is chosen for training. This improves the possibility of finding rules that are associated with small group of values during training. Another approach is to choose different number of PR data as training sets. We use three different training data sets. The first data set (Case 1, Table 1) choses 1224 PRs from only one software project. The second one (Case 2) builds equal distributed value for a medium size of 3400 PRs (1000 PRs from each value of '*Class*') from all software projects. The third data set (Case 3) contains a large size of 5381 PRs from all software projects.

We use two learning engines to discover rules from the PR data set– single support CBA (labelled a in the Table 1 e.g., Case1a) and multiple support CBA (labelled b in

the Table 1 e.g., Case1b). Constraints, support and confidence, are included in rules to control the quality of results. Confidence is the measure of the strength of a rule that indicates the probability of having consequence(s) in the rules provided that the rule contains certain antecedent(s). Support indicates the number of input data supporting the rule. Since, users are interested in rules with worth consideration or more preferred or more certain, a threshold for support and confidence is set.

Setting a threshold for minimum support and confidence is a result of trial and error. If these factors are set too high, very few rules may be discovered. If the factors are set too low, too many rules may be generated with very low values. Under certain situation, attributes in the data are not likely to have uniform distributions, and many attributes are of very low frequency. Therefore a single support for all attributes may not be able to discover important rules that involve such rare attributes (very low frequency). This problem is relieved by setting multiple supports that allow user to choose different minimum supports to different attributes. Table 1 reports the classification mining results on all three cases and the associative rule mining results as Case4. The test data is 10% of the whole data set and chosen randomly with the special consideration that each output class is representing in the test data.

Table 1: CBA Mining Results Summary. Rules are ranked by confidence.

	#Rules	Error rate (%)		Time cost (seconds)	
		Training	Testing	Training	Testing
Case1a	10	45.180	47.56	1.01	0.07
Case1b	9	45.180	47.56	1.04	0.09
Case2a	41	57.04	51.95	0.41	1.1
Case2b	21	59.10	58.25	0.44	1.0
Case3a	20	43.51	43.5	2.2	2.0
Case3b	15	46.5	45.1	1.6	1.9
Case4a	10	45.180	N/A	0.66	N/A
Case4b	9	45.180	N/A	1.04	N/A

In general, all classification data mining operations in CBA software achieve around 46% Error rate in training data set (the lowest is 43.51%, the highest is above 59.10%). Above 51% correct prediction rate is achieved in testing data set (the lowest has 43.51%, the highest has 58.25%). Another interesting point is that the attempt to improve the accurate prediction in the way of equal-distributed target-value samples does not lead much change; there is only roughly 3% improvement over the final result. The error rates from using multiple supports are higher and the number of extracted rules is lower than those from using single support mining engine. There is no rule that has confidence value larger than 80%, however they do describe some characters of the PR fixing patterns. Therefore they are useful for software project management in estimation bug fixing related time issues.

Followings are examples of generated classification rules with CBA:

Rule 1: If *severity= non-critical and Time-to-fix = 3 to 30 days and priority= medium*
Then *class = doc-bug.* Confidence = 82.7%, Support = 2.7%

Rule 2: If *severity= critical and Time-to-fix = less than 3 days and priority = high* Then *class = sw-bug.* Confidence = 75.2%, Support = 2.3%

Overall the extracted rules conclude that software related bugs can be fixed within 3 days with above 75% confidence if they have high priority and are in critical condition. It may take 3 months to fix the problem if the corresponding priority and severity are graded as medium and serious. Certainly, the confidences of the rules are low, and only a small number of cases support these rules.

Table 2: C5 Mining Results Summary

	Normal mining		Mining with Boosting		Mining with cross-validation (10-fold)	
	Training	Testing	Training	Testing	Training	Testing
#Rules	51	11	N/A.	N/A	57.7	12.4
Error Rate (%) (Rules)	41.5	42.6	41.3	42.6	43.9	42.8
Error Rate (%) (Trees)	40.3	42.5	39.4	42.6	44.1	43.1
Size of tree	141	21	N/A.	N/A	121.9	17.4
Process Time (seconds)	5.6	0.2	37.7	0.4	41.1	1.1

The software C5 was also used to perform classification data mining with the boosting and cross validation techniques (Table 2). The cross validation technique splits the whole data set into several subsets (called folds). Let each fold to be the test case and the rest as training sets in turn during training. Boosting is a technique for generating and combining multiple classifiers to give improved predictive accuracy. After a number of trials, several different decision trees or rule sets are combined to reduce error rate for prediction. Boosting takes a longer time to produce the final classifier, and may not always achieve better results than a single classifier approach does, especially when the training data set has noise. Boosting and cross validation techniques do not generate a new rule, but try to find a better rule from the existing results. They only produce better results than the individual trees if the individual trees disagree with one another. Some example extracted classification rules with C5 are:

Rule 1: When a PR is in *low priority* and the *time spent is around half a day (0.5 day)* Then the rule has a high probability (87.5% Confidence) to classify a bug to be a *document related bug*.

Rule 2: When a PR is in *medium priority* with *non-critical severity* and the *time spent is around 1.1 day* Then the rule has 84.6% Confidence to classify a bug to be a *document related bug*.

Rule 3: When a PR is in *low priority* and the *time spent for fixing is around 1 week* Then the rule has 83.3% Confidence to classify a bug to be a *software bug*.

In general, all data mining operations achieves around 42% error rate in rules from the training set (the lowest is 40.3%, the highest is 43.9%). Similar error rate value is

achieved for the generated trees in testing data set (the lowest is 39.4%, the highest is 44.1%). Both of the rates are better than CBA results. The time efficiency of C5 is also better than CBA.

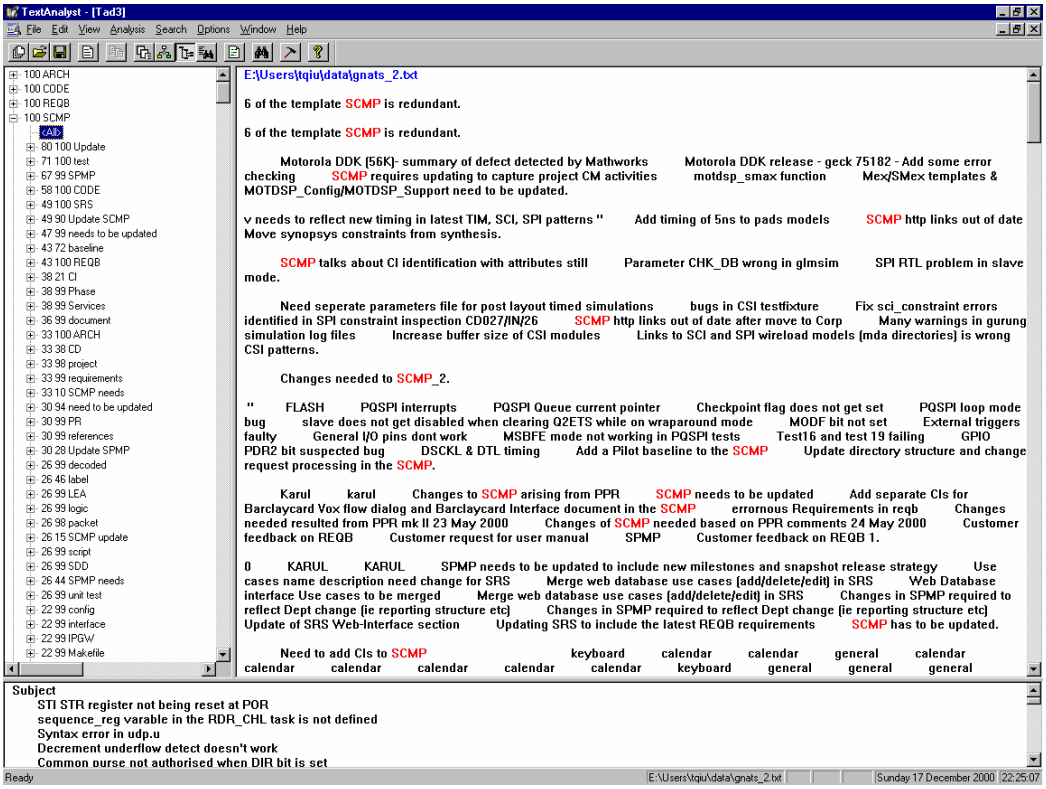

Figure 3: Text Analyst Mining: An interface

5.2 Text Mining in PR data

In order to find valuable knowledge from thousands lines of text, we categorise the pure text into several document types based on certain background knowledge. The analysis of the text together with the rules obtained from classification and association can more accurately predict the time and cost of fixing PRs. We used TextAnalyst [3] tool to automatically summarise the pure text data and extract some valuable rules. It builds up a semantic network for the investigation over the PR data. Each element of the semantic network is characterised by a weight value and a set of relationship of this element to other elements in the network. Every relationship between elements is also assigned a weight value. The semantic network can then provide a concise and accurate summary of the analysed text.

TextAnalyst automatically creates the semantic network based on the structure, vocabulary and volume of the analysed text, without any predefined rules. The semantic network tree of the PR data (figure 3) contains a set of the most important words or word combinations, called *concepts*. The relations among those concepts

together with the semantic weights of concepts and relations are also shown. The values of the weights range from 0 to 100, which correspond to the probability that the associated concept is characteristic for the whole PR data. It also shows all the text related to a concept if the user clicks the corresponding concept.

Text mining is applied on a total 11226 cases. An interesting result is obtained for SMTP, Software Configuration Management Plan, a support document in every MASC project. Since SCMP is not a main design document for a project with just about tens pages, it has never been considered as a trouble making item. But amazingly, the result showed that SCMP has 71 percentage probability of appearing in test related PR records, and 58 percentage probability of appearing in Code related PR records. This result is even higher than the result associated with SRS ("Software Requirements Specification", a main development document directly related to software). Although we can not say SCMP causes more problems than SRS, but the higher appearance of SCMP inside test related PR definitely shows a warning. It is worth for software engineers to be more careful when dealing with SCMP document, and hence reducing the total cost of fixing SCMP related PRs. Another analysis shows that a test related PR has even a higher weight (36, 100) in document related bugs than a SRS related PR (35, 99) does. Which suggests a better project management should not only focus on the quality of product related documents, but also pay attention on the quality of testing related documents.

5.3 Existing problems in performing mining

The error rates of testing data sets in both CBA and C5 are higher than expected. Although several approaches are attempted to reduce the error such as uniform distribution of values, cross validation, boosting, different size of training set, etc. Unfortunately, the average error rate is only fallen down by 5% from 47% to 42%. The best result is 9% decrease from 46% to 37%. These results indicate that some amount of noise is still existent in data after dealing with the noise during pre-processing.

For example, the relationship between PRs and human resources within a particular project plays a great impact. The time needed to fix a bug is different for each project depending upon the actual human resources available. We have used only the attribute '*Responsible'* to indicate the human resource available. Truly, the relationship with the human resources available for past projects whose data was analysed is needed to use time patterns to help project leaders to predict time consummations more accurately. The use of additional data source 'Change Request data set' that records all customer request process data may rectify this problem.

Another reason is a non-uniform value distribution. For example, there are only 342 PRs with *change-request* value in the data set, compared to more than 5900 PRs related to *sw-bug*. Any potential rule associated to *change-request* can be heavily affected due to the presence of large size group with other values. Again the use of additional data sources together with the PR data set can rectify this problem. We also attempted to use neural network techniques, but it is difficult to interpret the outputs.

6 Future Directions and Conclusion

This paper explored the use of data mining techniques on a set of data collected from the software engineering process under a real software business environment. Some useful rules are inferred on the time patterns of the PR fixing and the relationship between the content and the type of a PR in the form of association rules, classification rules or semantic trees.

The time patterns rules may help a project leader to estimate or predict time consummations more accurately than before. Another finding suggests that bug fixing efforts have more probabilities to be spent on test related PR. This could cost the project team a lot of time in fixing non-product-related problems. By giving more attention in design and development of these documents, the project efficiency will be improved.

Results of the application indicate that data mining techniques bring more power to improve the quality and efficiency of the software development process, even though the scale of the data mining task is limited. It will be interesting to apply data mining to different phases of software development such as software quality data, etc. As in many other domains, the benefits and capabilities brought by data mining in software engineering domain are worth of further investigations.

7 References

1. CBA, http://www.comp.nus.edu.sg/~dm2/
2. C5.0, http://www.rulequest.com/see5-info.html
3. TextMiner, http://www.megaputer.com/company/index.html
4. P.Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi, "*Discovering Data Mining: From Concept to Implementation*", Prentice Hall, 1997.
5. H. Edelstein, "*Mining for gold. (Selecting data mining tools)*", Information Week, April 21, 1997 n627 p53 (6).
6. J. Han and M. Kamber, "*Data mining: concepts and techniques*", Morgan Kaufmann Publishers, 2001.
7. C. Westphal, and T. Blaxton, "Data Mining Solutions: Methods and Tools for Solving Real-World Problems", John Wiley & Sons, Inc, 1998.